Temporal Logic with "Until", Functional Reactive Programming with Processes, and Concrete Process Categories

Wolfgang Jeltsch

TTÜ Küberneetika Instituut

The Seventh ACM SIGPLAN Workshop on
Programming Languages Meets Program Verification

22 January 2013

Functional reactive programming

- extension of functional programming
- allows programs to deal with temporal aspects in a declarative fashion
- two key concepts:
  - time-dependent type membership
  - special type constructors:
    - $\Box$ time-varying values
    - $\Diamond$ events
- Curry–Howard correspondence to temporal logic:
  - time-dependent trueness
  - special operators:
    - $\Box$ will always hold
    - $\Diamond$ will eventually hold

Categorical models of FRP

- ingredients:

  totally ordered set $(T, \leqslant)$ time scale

  CCCC $\mathcal{B}$ simple types and functions
- functor category $\mathcal{B}^T$ models FRP types and operations with indices denoting inhabitation times
- CCCC structure of $\mathcal{B}^T$ from CCCC structure of $\mathcal{B}$ with operations working pointwise
- functors $\Box$ and $\Diamond$ defined as follows:

$$(\Box A)(t) = \prod_{t' \geqslant t} A(t')$$

$$(\Diamond A)(t) = \coprod_{t' \geqslant t} A(t')$$

From "until" to processes

- more temporal operators from linear-time temporal logic:
  - $\triangleright'$ strong "until"
  - $\blacktriangleright'$ weak "until"
- semantics given by functors $\triangleright'$ and $\blacktriangleright'$:

$$(A \triangleright' B)(t) = \coprod_{t' \in (t,\infty)} \left(\left(\prod_{t'' \in [t,t')} A(t'') \right) \times B(t') \right)$$

$$(A \blacktriangleright' B)(t) = (A \triangleright' B)(t) + \prod_{t' \in [t,\infty)} A(t')$$

- FRP analogs of "until" proofs are processes:
  - normally finite-length time-varying value plus terminal event
  - in the case of $\blacktriangleright'$ also nontermination possible

Applications of processes

- stereo playback with different guarantees:

  $(\mathbb{R} \times \mathbb{R}) \blacktriangleright' 1$ none

  $(\mathbb{R} \times \mathbb{R}) \triangleright' 1$ termination

  $(\mathbb{R} \times \mathbb{R}) \blacktriangleright' 0$ nontermination

- stereo playback with additional information:

  $(\mathbb{R} \times \mathbb{R}) \blacktriangleright' (1 + 1)$ reason of termination (end of track vs. abort)

- alternating stereo/mono playback with different guarantees:

  $\nu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \blacktriangleright' \mathbb{R} \blacktriangleright' \alpha$ nontermination

  $\nu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \triangleright' \mathbb{R} \triangleright' \alpha$ switch, nontermination

  $\nu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \blacktriangleright' (1 + \mathbb{R} \blacktriangleright' (1 + \alpha))$ none

  $\nu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \triangleright' (1 + \mathbb{R} \triangleright' (1 + \alpha))$ switch

  $\mu\alpha \,.\, (\mathbb{R} \times \mathbb{R}) \triangleright' (1 + \mathbb{R} \triangleright' (1 + \alpha))$ termination

Processes as the core concept of FRP

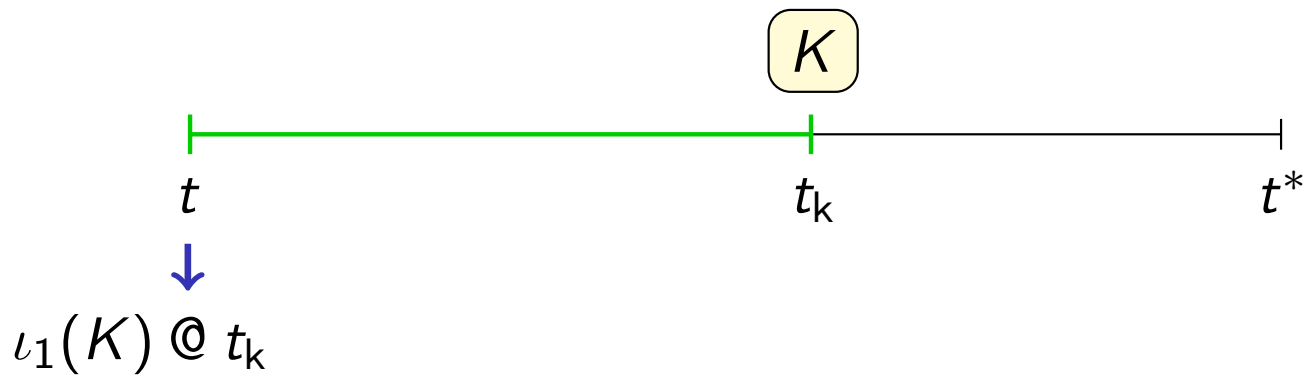
- introduction of processes increases expressiveness
- processes cover time-varying values and events as special cases:

$$\Box A \cong A \blacktriangleright' 0$$
$$\Diamond A \cong A + 1 \triangleright' A$$

An example program component

- looks for the next key press up to a certain timeout
- emits a value of type $\Diamond(\mathrm{Key} + 1)$ when it starts:

  Case 1 key press before timeout:



  Case 2 no key press before timeout:

A noncausal operation

- hypothetical polymorphic operation d from $\Diamond(\tau_1 + \tau_2)$ to $\Diamond\tau_1 + \Diamond\tau_2$:

$$\iota_1(x) \mathbin{@} t' \mapsto \iota_1(x \mathbin{@} t')$$
$$\iota_2(y) \mathbin{@} t' \mapsto \iota_2(y \mathbin{@} t')$$

- applying d to the output of the key press listener gives value of type $\Diamond\mathsf{Key} + \Diamond 1$:

  key press before timeout $\iota_1(K \mathbin{@} t_\mathsf{k})$
  no key press before timeout $\iota_2(\mathsf{tt} \mathbin{@} t^*)$

- tells us immediately if the user will press a key before the timeout
- so d cannot exist

Semantics allow for noncausal operations

- polymorphic operations from $\Diamond(\tau_1 + \tau_2)$ to $\Diamond\tau_1 + \Diamond\tau_2$ modeled by natural transformations τ with

$$\tau_{A,B} : \Diamond(A + B) \to \Diamond A + \Diamond B$$

- there is such a τ (which is even an isomorphism):

$$\coprod_{t' \geqslant t} \Big(A(t') + B(t') \Big) \cong \coprod_{t' \geqslant t} A(t') + \coprod_{t' \geqslant t} B(t')$$

- reason:

  semantics do not deal with time-dependent knowledge about values

Knowledge-aware semantics

- replace category $\mathcal{B}^T$ by category $\mathcal{B}^I$ where

$$I = \{(t, t_o) \in T \times T \mid t \leqslant t_o\}$$

- dealing with knowledge at t_o:
  - objects $A(t, t_o)$ deal with knowledge at t_o
  - morphisms $f_{(t,t_o)}$ transform knowledge
- $(A \blacktriangleright' B)(t, t_o)$ defined as follows:

$$\coprod_{t' \in (t,t_o]} \left(\left(\prod_{t'' \in [t,t')} A\left(t'', t_o\right) \right) \times B\left(t', t_o\right) \right) + \prod_{t' \in [t,t_o]} A\left(t', t_o\right)$$

Compatibility of knowledge transformations

- knowledge transformations $f(t, t_o)$ and $f(t, t'_o)$ must be compatible
- extend set I to category $\mathcal{I}$ by adding morphisms

$$(t, t_o, t'_o) : (t, t'_o) \to (t, t_o)$$

for $t \leqslant t_o \leqslant t'_o$

- modeling of knowledge reduction:

$$A(t, t_o, t'_o) : A(t, t'_o) \to A(t, t_o)$$

- morphisms of $\mathcal{B}^{\mathcal{I}}$ are natural transformations:

$$\begin{array}{ccc} A(t, t_o) & \xleftarrow{A(t, t_o, t'_o)} & A(t, t'_o) \\ \Big\downarrow{\scriptstyle f_{(t,t_o)}} & & \Big\downarrow{\scriptstyle f_{(t,t'_o)}} \\ B(t, t_o) & \xleftarrow[B(t, t_o, t'_o)]{} & B(t, t'_o) \end{array}$$

Upper bounds for occurrence times

- definition of functor $\rhd'$ not directly possible
- introduction of new functor $\rhd'_{\_} : \mathcal{T} \to \left(\mathcal{B}^{\mathcal{I}}\right)^{\mathcal{B}^{\mathcal{I}} \times \mathcal{B}^{\mathcal{I}}}$ where $\mathcal{T}$ is the category of $(T, \leqslant)$
- $\rhd'_{t_\mathrm{b}}$ models a process type constructor with upper bound t_b for termination time
- $(A \rhd'_{t_\mathrm{b}} B)(t, t_\mathrm{o})$ defined as follows:

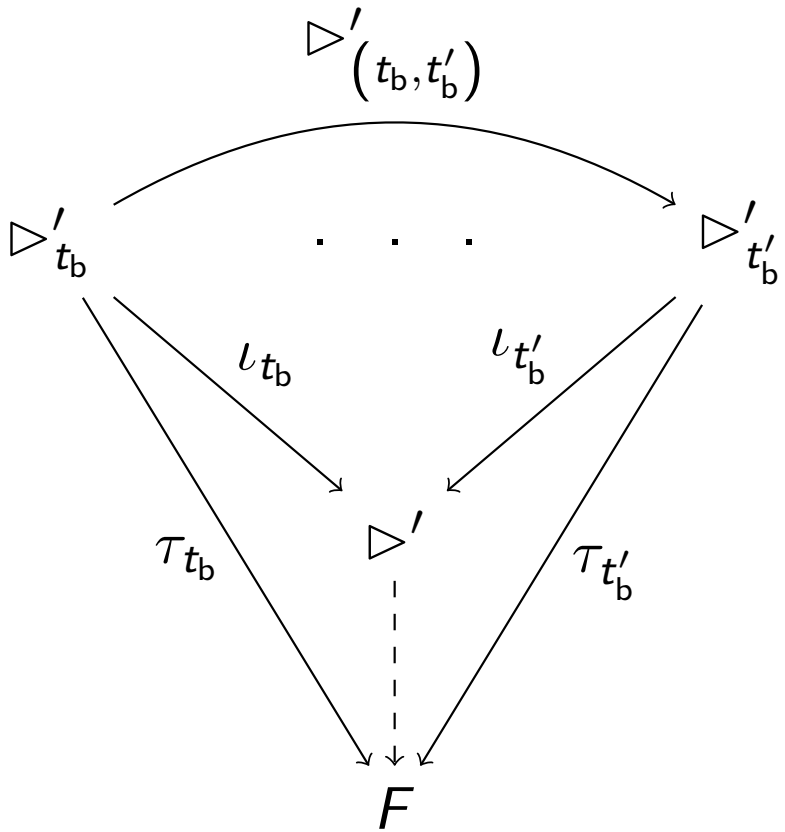
$$\begin{cases} 0 & \text{if } t_\mathrm{b} < t \\ \coprod_{t' \in (t, t_\mathrm{b}]} \left(\left(\prod_{t'' \in [t, t')} A(t'', t_\mathrm{o}) \right) \times B(t', t_\mathrm{o}) \right) & \text{if } t \leqslant t_\mathrm{b} \leqslant t_\mathrm{o} \\ (A \blacktriangleright' B)(t, t_\mathrm{o}) & \text{it } t_\mathrm{o} < t_\mathrm{b} \end{cases}$$

- modeling of type conversion:

$$\rhd'_{\left(t_\mathrm{b}, t'_\mathrm{b}\right)} : \rhd'_{t_\mathrm{b}} \to \rhd'_{t'_\mathrm{b}}$$

Definition of the $\triangleright'$-functor

- type constructor $\triangleright'$ is the least upper bound of all $\triangleright'_{t_b}$-constructors
- functor $\triangleright'$ must be a colimit of the functor $\triangleright'\_$:



The shape of the $\triangleright'$-functor

Theorem

If $(T, \leqslant)$ has a maximum $t_{\max}$, then $\triangleright' \cong \triangleright'_{t_{\max}}$.

Theorem

If $(T, \leqslant)$ has no maximum, then $\triangleright' \cong \blacktriangleright'$.

Causality ensured

Theorem

There are concrete process categories that do not contain any natural transformation τ with

$$\tau_{A,B} : \Diamond(A + B) \to \Diamond A + \Diamond B \quad .$$

Conclusions

- processes:
  - result of extending the Curry–Howard correspondence between FRP and temporal logic to cover "until" operators
  - make FRP more expressive
  - generalize time-varying values and events nicely
- knowledge-aware categorical models:
  - express causality of FRP operations
  - cannot express liveness constraint of $\triangleright'$ for unbounded time
- ultimate goal is an axiomatic semantics with the following properties:
  - expresses causality
  - expresses liveness constraint of $\triangleright'$ in general
  - covers concrete process categories as a special case